Information Technology

Information Technology and Communications (ITCom)

FINAL PROJECT

IMPLEMENTING NX REMOTE DESKTOP TECHNOLOGY IN THE LTSP SYSTEM

Author: Simo Poskiparta
Instructor: Ilpo Kuivanen
Supervisor: Ari Pantsar

Approved 17.4.2007

Ilpo Kuivanen
Principal Lecturer

PREFACE

When I started this project in the start of the year 2007, three months seemed to be more than enough for studying the NX Remote Desktop technology and writing this study. During the time it came very clear, that projects like these are much more consuming and stressing than predicted.

At first I would like to thank Finnish Meteorological Institute Technical Services workers for providing me this final project subject. When information was needed during the project, people were happy to help. Special thanks to Ari Pantsar, who instructed me many times during this process. Thanks also to lecturer Ilpo Kuivanen for supervising this project.

Thanks go also to rock music for maintaining my sanity during this project.

15.4.2007 Simo Poskiparta

STADIA
HELSINKI POLYTECHNIC

ABSTRACT

Name: Simo Poskiparta

Title: Implementing NX Remote Desktop Technology in the LTSP System

Date: 15.4.2007 Number of pages: 39

Department: Information Technology Study Programme: Information Technology and Communication

Instructor: Analyst Ari Pantsar, Finnish Meteorological Institute

Supervisor: Lecturer Ilpo Kuivanen, Stadia

This project focuses on studying and testing the benefits of the NX Remote Desktop technology in administrative use for Finnish Meteorological Institutes existing Linux Terminal Service Project environment. This was done due to the criticality of the system caused by growing number of users as the Linux Terminal Service Project system expands. Although many of the supporting tasks can be done via Secure Shell connection, testing graphical programs or desktop behaviour in such a way is impossible.

At first basic technologies behind the NX Remote Desktop were studied, and after that started the testing of two possible programs, FreeNX and NoMachine NX server. Testing the functionality and bandwidth demands were first done in a closed local area network, and results were studied. The better candidate was then installed in a virtual server simulating actual Linux Terminal Service Project server at Finnish Meteorological Institute and connection from Internet was tested to see was there any problems with firewalls and security policies. The results are reported in this study.

Studying and testing the two different candidates of NX Remote Desktop showed, that NoMachine NX Server provides better customer support and documentation. Security aspects of the Finnish Meteorological Institute had also to be considered, and since updates along with the new developing tools are announced in next version of the program, this version was the choice. Studies also show that even NoMachine promises a swift connection over an average of 20Kbit/s bandwidth, at least double of that is needed.

This project gives an overview of available remote desktop products along their benefits. NX Remote Desktop technology is studied, and installation instructions are included. Testing is done in both, closed and the actual environment and problems and suggestions are studied and analyzed. The installation to the actual LTSP server is not yet made, but a virtual server is put up in the same place in the view of network topology. This ensures, that if the administrators are satisfied with the system, installation and setting up the system will go as described in this report.

Keywords: remote desktop, NX Remote Desktop, FreeNX,

STADIA
HELSINKI POLYTECHNIC

TIIVISTELMÄ

Tekijä: Simo Poskiparta

Työn nimi: Implementing NX Remote Desktop Technology in the LTSP System

Päivämäärä: 15.4.2007	Sivumäärä: 39 s.
Koulutusohjelma: Tietotekniikka	Suuntautumisvaihtoehto: Information Technology and Communication

Työn ohjaaja: suunnittelija Ari Pantsar, Ilmatieteen laitos

Työn valvoja: lehtori Ilpo Kuivanen, Stadia

Tässä insinöörityössä tutkittiin ja testattiin mahdollisuutta ottaa käyttöön NX Remote Desktop -ohjelmisto Ilmatieteen laitoksen Linux Terminal Service Project -järjestelmän etähallintaa varten. Käyttäjien lisääntyessä järjestelmän kriittisyys kasvaa, joka lisää myös hallinnallisia toimia. Vaikka suurimman osan hallinnasta pystyy tekemään Secure Shell -konsoliohjelmalla, on graafisten ohjelmien sekä työpöytäympäristön testaaminen sillä mahdotonta.

Aluksi tarkasteltiin tekniikoita, joihin NX Remote Desktop -teknologia perustuu, ja tämän jälkeen testattiin kahta eri tuotetta, NoMachine-yhtiön NX Remote Desktop Server -ohjelmaa sekä vapaaseen lähdekoodiin perustuvaa FreeNX Server -ohjelmaa. Toiminnallisuutta ja kaistavaatimuksia testattiin aluksi suljetussa verkossa ja tuloksia analysoitiin. Paremmaksi havaittu ohjelma asennettiin Ilmatieteen laitoksen virtuaalipalvelimelle, jonka tarkoituksena oli simuloida varsinaista Linux Terminal Service Project -palvelinta. Tämän jälkeen käytettiin ulkoista internet-yhteyttä yhteyden muodostamiseen, jotta nähtiin tuottavatko ulkoiset ja sisäiset palomuurit ongelmia.

Tutkimukset osoittivat NoMachine-yhtiön tarjoavan paremmat dokumentaatiot sekä käyttäjätuen. NoMachine päivittää ohjelmiaan usein, ja testausvaiheessa on myös monta ohjelman hallintaa parantavaa lisäohjelmaa. Koska tietoturvallisuus on oleellinen asia myös Ilmatieteen laitoksen toimintamallissa, päädyttiin työssä käyttämään NoMachine NX Remote Desktop -palvelinta FreeNX-palvelimen sijaan. Vaikka NoMachine verkkosivuillaan lupaa ohjelman toimivan keskimäärin 20 Kbit/s kaistavaatimuksella, osoittivat testit, että kaistaa tarvitaan odellisuudessa mukavaan käytettävyyteen vähintään kaksi kertaa enemmän.
Tässä työssä esitellään aluksi etätyöpöydän käytön peruskonseptia ja sen jälkeen tutustutaan tarkemmin NX Remote Desktop -teknologiaan ja olemassaoleviin tekniikoihin, joihin se perustuu. Työssä esitetään myös ohjelman asennusohjeet, ja miten yhteys muodostetaan sekä sisä- että ulkoverkosta. Asennusta varsinaiselle Linux Terminal Service Project -tuotantopalvelimelle ei vielä tehdä, mutta virtuaalipalvelin vastaa verkkotopologian mukaan kyseistä palvelinta. Tämä mahdollistaa myöhemmän käyttöönoton tuotantopalvelimella työssä esitetyllä tavalla.

Avainsanat: etäkäyttö, NX Remote Desktop, FreeNX

PREFACE

ABSTRACT

TIIVISTELMÄ

TABLE OF CONTENTS

LIST OF ACRONYMS

LIST OF ACRONYMS

ADSL	Asymmetric Digital Subscriber Line
CDE	Common Desktop Environment
CUPS	Common Unix Printing System
FMI	Finnish Meteorological Institute
GDM	GNOME Display Manager
GPG	Gnu Privacy Guard
GPL	General Public Lisence
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HTTP	Hypertext Transfer Procotol
ICA	Independent Computing Architecture
ICT	Information and Communication Technology
IPsec	Internet Protocol security
IPv6	Internet Protocol version 6
KDE	K Desktop Environment
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LTSP	Linux Terminal Server Project
PAM	Pluggable Authentication Modules
PXE	Preboot Execution Environment
RDP	Remote Desktop Protocol
RFB	Remote FrameBuffer
SMB	Server Message Block
SSH	Secure Shell terminal program

SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
VNC	Virtual Network Computing
VPN	Virtual Private Network
XDM	X Window Display Manager
XDMCP	X-Display Manager Control Protocol
X11or X	X-Window System

1. INTRODUCTION

The focus on this project is to test and implement a remote desktop program NX Remote Desktop. The project is divided in two parts. At first, testing of two of the possible programs operating with NX Remote Desktop technology. First one is a open source program FreeNX, developed by Fabian Franz and the other is commercial program from NoMachine company. Both of these are tested in a closed local area network and results are analysed. Advantages and disadvantages of both versions are studied, and more suitable one will be installed in Finnish Meteorological Institute's (Finnish Meteorological Institute) existing Linux Terminal Service Project LTSP system.

The characteristics that will be observed over others are security and operability. This is because Finnish Meteorological Institute requires a certain amount of security, and even though ease of useness is not mandatory, user friendly systems tend to endure their existence.

At first a brief look to overall remote desktop productions is made. Benefits and basics of telecommuting are described. After that two possible NX Remote Desktop programs from two different authors are studied following by more detailed view into technology around NX Remote Desktop. Installation of server, node, and client will be covered closely, and after testing in a closed environment is done results will be examined. The final setting is, if possible, to install NX Remote Desktop program in LTSP system for administrative use and test the operability in Intranet and through SSH tunneling from Internet.

A remote desktop is a way to use computers with a graphical interface over the network from anywhere in the world. It is done in a way that the server computer sends the necessary data to display the graphic to the client

computer. In modern working conditions, e.g. working from home or abroad has became more common. The remote desktop is a functional way to access the needed computer from far away, of even manage servers over the network, in case that they need a Graphical User Interface (GUI) to do it.

1.1. Finnish Meteorological Institute

The predecessor of the modern Finnish Meteorological Institute was called the Magnetic Observatory of the Helsinki University, founded on 1838 by emperor Nikolai I. After the Magnetic Observatory of the Helsinki University was moved from the control of the university to the Finnish Science Association, the name was also changed to a Main Institute of the Meteorology in 1881. [1]

In 2005 the new building for Finnish Meteorological Institute was ready, and all the working personnel from previous buildings, Kaisaniemi and Herttoniemi were moved to a brand new Dynamicum building in Kumpula. This was great improvement, since older buildings capacity was limited, and it was time to upgrade the technological means also. This also eliminated the need for IT personnel to run around four separate buildings fixing computers and helping people in their problems. The new building provides also a great new up-to-date image to all the institute and improves working conditions of all personnel.

1.2. Finnish Meteorological Institute Technical Services

Finnish Meteorological Institutes Technical Services employees almost 150 persons. It is a responsible for providing research material and observations of the Finnish atmosphere. The whole bunch is divided in three separate sections, Observation Services, ICT Services, and Information System

Services. The main focus in all of the sections is to provide means for meteorological research and to maintain operability under any circumstances.

The Observation Services takes care of observation equipment, obtains observations from their observation stations and develops and maintains the weather radar network. It also provides expert services for developing projects in both homeland and abroad. [2]

ICT Services takes care of the functionality of the network infrastructure, and provides helpdesk services to workers. It is responsible for planning and monitoring the security of the network as well as taking care of the purchases of new computers and servers. To meet the growing demand of the weather pattern calculation Finnish Meteorological Institute bought a new supercomputer at late 2005.

The most important task of the Information System Services is to preserve and to process observation and research data in a usable format for the meteorological use.

1.3. Linux Terminal Service Project Environment

All the workers in Finnish Meteorological Institute have at least one computer, many of the people have two even though only one is usually needed. This of course leads to a remarkable power consumption that can be seen as growing electricity bills. In December 2005 the first tests with Linux Terminal Service Project started. The main idea was to replace as many as possible of the computers with thin clients. Since these thin clients holds no moving parts, and start their applications from the server, the power consumption decreases and if the client is broken a new one can just be wired and its ready to use. Since Linux is the second most used operating system in Finnish Meteorological Institute, the replacing of the old Linux computers with thin clients could be started.

Linux Terminal Service Project (LTSP) is a client-server oriented thin client system founded on 1999 in the USA. The existing architecture is quite simple, consisting of couple of servers, a bunch of clients, and LTSP intranet. As the client holds no starting media, it queries the necessary files from the LTSP server through its Preboot Execution Environment (PXE) starting chip. Thin as the client is, it differs from so called dumb terminals in a way that LTSP client may run some programs locally, as the dumb terminals start every program from the server. LTSP client holds no moving parts, and is silent and power saving. The existing system is put together with Finnish company Opinsys Oy, which also provides the necessary terminals.

The network is divided in two parts, encrypted and unencrypted. As the terminal starts, it uses the unencrypted network address to fetch its starting files. All the user X-Display manager traffic is encrypted with an IPsec layer above the TCP/IP layer, implemented with Openswan, which is most used IPsec application for Linux. This also provides user authentication and verifies data. The clients have also VIA PadLock chip to enable fast encryption and decryption in a hardware basis. As far as it is known, the LTSP environment in Finnish Meteorological Institute is one of its kind in the whole world.[3] The structure of the LTSP system in Finnish Meteorological Institute can be seen on *figure 1*.

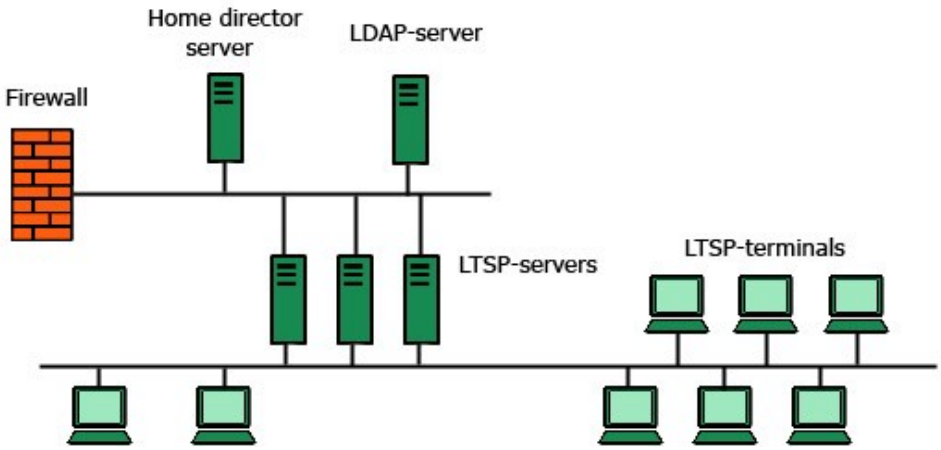

Figure 1: The LTSP network [4]

In the spring 2007 there are three LTSP servers that run Ubuntu 6.06. There is also home directory server that holds user data, and Lightweight Directory Access Protocol (LDAP) server that takes care of the authentication. The LTSP environment is accomplished with three HP ProLiant BL20p blade servers running Ubuntu 6.0.6 Linux.

1.4. Objectives

The objective of this project is first to test and to install the NX Remote Desktop system to the Finnish Meteorological Institute existing LTSP environment. This is done by first testing out which one of the server candidates, whether the FreeNX or NoMachine NX Free Edition should be used. First both are installed and tested in an closed environment, and after that better program is installed in the server simulating the actual LTSP server, and tested in a real situation. Benchmarks of the network usage are also to be included, as well as interviews with the administrators. Since the network security policies are very strict, testing is done by forming a SSH tunnel through a firewall between the server and the client, thus disabling the need for opening new ports to firewall which could lead to a security threat.

Basics of the remote desktop use including benefits and secure threats are discussed below.

2. REMOTE DESKTOP

This section describes the basic character of remote desktop. Benefits and security threats are also examined. Since there are many ways to achieve remote desktop connection, some operating systems have an build-in mechanism for remote desktop usage, such as Terminal Services Client (later known as Remote Desktop Client) in Windows XP. There are also a wide range of products to establish a connection between the client and the server machine. The most important aspect when using remote desktops is of course security. Usually the companies' security policies determines the possibilities of using remote desktop connections, or what kind of tasks can be done with it. The policies may dictate that password and user management, billing, and such are only done locally, not over the possible insecure network.

2.1. Benefits ot the Remote Desktop

The benefits of the remote desktop system are considerable. If an administrator needs to maintain multiple servers or end-user workstations, and if basic terminal connection over SSH is not enough, the administrator needs to go where the computer physically stands. In a case of the basic helpdesk jobs, the problems are usually desktop oriented. This means that the administrator has to leave the desk to go fix the problem. Following picture, *figure 2* shows the basic idea of using remote desktop.

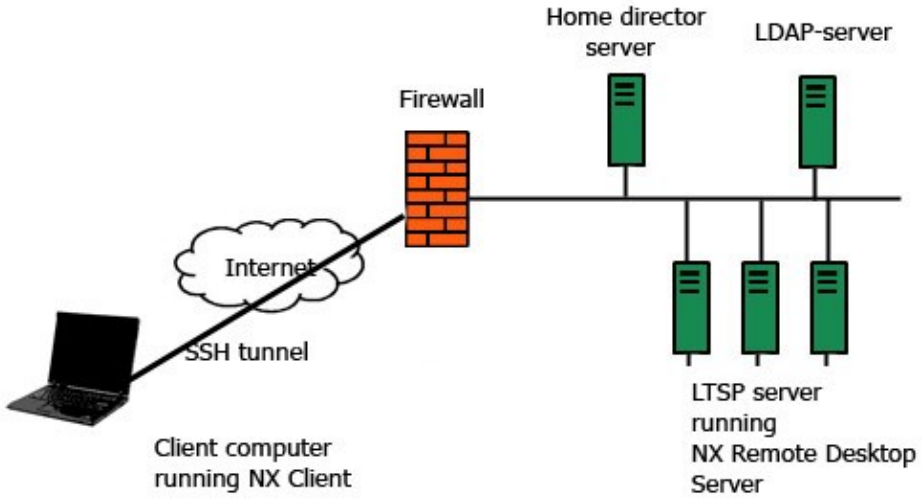

Figure 2: Using remote desktop

The picture shows that a worker who has access to the Internet can put up a SSH tunnel to company's firewall thus granting access to company's Intranet. Remote desktop eliminates this to and fro running. If employees have problems with e-mail program e.g. the administrator might establish a remote desktop connection to the users workstation, and fix the problem without leaving the desk. User management can also be maintained with a GUI program, thus placing the server in the secure computer hall means more running. If the remote desktop connection could be established between administrators and the server computer, the time saved means more money. The situations described above needs of course a program to form such a connection, sufficient bandwidth allocation, and up-to-date security policies.

The bigger companies that provide helpdesk services to hundreds or thousands of people usually use some kind of a remote desktop system. This makes it possible to centralize the helpdesk staff to a one building, whereas people calling to helpdesk may reside anywhere on the world.

2.2. Security Threats

The basic idea is usually that there are no secure systems. Expecially in a situation where data is sent over the network, there are always possibilities for a security threats. Even with a secured connection, there could be weaker links somewhere else in the system. Since the NX Remote Desktop relies on SSH technology, the most popular SSH threats are described below.

When first establishing a connection with SSH between the two hosts, the first thing is to trade encryption algorithm key pair. This is done via relatively weak encryption, and if there is a man-in-the-middle attack, the attacker can read the incoming connections between the hosts.[5] This can be eliminated by moving the public/secret key pair e.g. in a pocket if possible.

Too often people leave the key to the server in a computer where they last formed a connection. After this the attacker can put up a fraud server and steal passwords and such from the people using the same server. It is strongly recommended to use SSH-2, the newer version of the SSH instead of the SSH-1 which has inherent design flaws.

There are also other security threats, such as user passwords. In many cases users have too weak passwords, and those can be hacked by brute force. In case of administrators this is not so often, but ageing passwords form another problem. It is widely recommended to change at least the most important passwords occasionally thus eliminating the risk of former workers knowing the actual passwords. The most important security aspect in this project is the use of secure SSH tunnel to establish a connection to the server from the Internet and to use the client program's option to use Secure Socket Layers (SSL) encryption on all traffic.

2.3. Overview of the Available Products

There are many products from different vendors for a remote desktop connection. The Windows introduced its own Terminal Services in Windows NT 4.0, and has developed it over the time to fit the needs of the Windows Servers. There are also a Remote Assistance option in Windows XP which makes it easy for workers to help one another, but the full Remote Desktop needs a Proffessional Edition or Media Center Edition 2005 or earlier. Citrix Presentation Server (formerly known as Citrix Metaframe) is another solution for remote desktop environment. It can run most of the Microsoft Windows programs, as well as Unix programs from the server.[6] Citrix Presentation Server relies on a Independent Computing Architecture (ICA) client to enable users to launch programs from the server machine either one program at a time or even the whole desktop at once. ICA is a proprietary protocol designed by Citrix Systems that passes data between the server and clients.

Virtual Network Computing (VNC) is also a possibility for remote desktop use. It is a platform independent tunneling system, and relies on a Remote FrameBuffer (RFB) protocol. User can set up VPN tunnel between home office and business office and transfer data securely encrypted over the pipe. Keyboard presses and mouse clicks are sent to server from a client computer, and screen updates are sent back. VNC is not a secure protocol by default, but it can be tunneled over SSH or Virtual Private Network (VPN) connection thus increasing the security. The basic way to send graphic data is to tell client machine to draw a pixel in a specified x, y position. This is done through the framebuffer, and consumes a lot of bandwidth. To overcome the bandwidth demands, there are several encoding techniques available. However, the NX is quite similar to the VNC but the bandwidth demands of the NX Remote Desktop are promised to be considerable smaller.

3. NX TECHNOLOGY

This section introduces the technologies which are used in the NX Remote Desktop program. The modules of the program are also desrcibed. Since the NX Remote Desktop is based on some of the existing protocols such as SSH and X Window System, they are also examined briefly. SSH is a secure way to log on to remote computers, or for executing commands remotely.It was developed due to an unsecure features of the predecessors. The X Window System is a commonly known display and networking protocol. X Window System is not just a display manager, it is a client-server system which makes it meet the demands of the remote desktop environment. There is also a possibility to use a graphic manager to tune the connections and user configurations of the NX remote desktop. This relies on the usage of Apache HTTP web server to set up a web page based graphical administration interface. However the detailed information of Apache architecture is not discussed here, as the graphic manager is still at the beta state and is not covered in this project.

NX technology is a way of establishing a remote desktop connection over a low bandwidth. NX remote desktop uses SSH protocol to send the data over a network and SSH public key cryptography for authentication. The basic modules and their operability is covered below.

3.1. X-Window System

The display manager is responsible for importing the graphical data to users display. Normally the X-server is started automatically, but the user may want to disable this feature, and start it from the text console. The original display manager is called X display manager (XDM), but there are also

modern managers such as the GNOME display manager (GDM) and the KDE display manager (KDE). The client/server display model provides a great flexibility in many environments, making it possible to run the applications from separate computer and sending the display data over network in a wanted display. It is recommended to use SSH tunnel to send display data over the internet because of the weak security level of X system. This can be also done in a local network depending on the security regulations. One of the benefits of the X11 is that it sends graphic as a vectors over the network. This eliminates need for a raw bitmaps to be sent over a congested line.

3.2. SSH

The Secure Shell (SSH) system is a secure way for transmitting data over network. It is a modern replacement of rlogin, rcp, rsh and telnet which were previous, more insecure programs. The SSH was developed due to insecure features of the predecessors.[7, p.697] It encrypts all communications between the two hosts, and is designed to withstand a large number of varying attacks. SSH uses public-key cryptography for authentication. This is done with a pair of keys, a public key and a private key. The public key is distributed, but the private key is kept secret. The authentication is done by comparing the private key to the public key mathematically. The next picture, *figure 3* explains the basic SSH public key encryption.

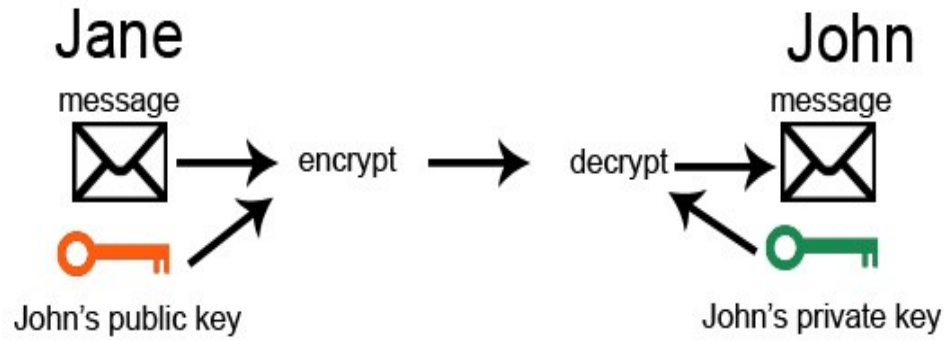

Figure 3: The public key encryption

Figure 3 shows the most common analogy to this kind of crypting. It is a locked mailbox with a mail slot. Anyone knowing the street address can go and drop a written message in it, but only the owner of the box has the key to open the slot and read message.

SSH is divided to three layers, the transport layer, the user authentication layer, and the connection layer.[8]

The transport layer is responsible for establishing a secure connection between the server and the host by exchanging the keys, and encrypting and decrypting the data. It can also provide data compression.

The user authentication layer, as the name says, is responsible for authenticating the user. This is done usually by username and password, but other methods for authentication can also be configured.

After a successful authentication multiple channels are opened by multiplexing the single connection between the two hosts. Each of these channels handles a communication for a different terminal session, for

example forwarded X11 information.

The SSH can be used to secure transfer of data, but also to tunnel traffic from one destination to other. Note that the DISPLAY variable and authentication information are automatically set up by SSH. The display number of the client starts at :10.0 and is incremented for each SSH connection that is forwarding X traffic. The procedure is shown in *figure 4* below.

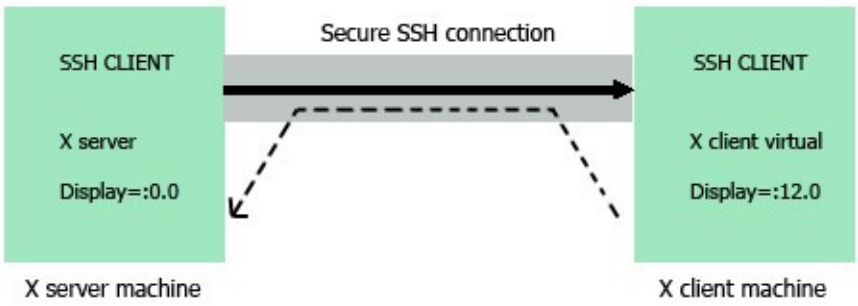

Figure 4: The SSH tunnel[9, p.747]

The figure shows how X-traffic is forwarded from server to client. SSH tunnel is made between the server and client to ensure secure data transfer.

3.3. NX Server

NX Server is a base module of the NX Remote Desktop system. The Server is installed in a computer with supported operating system. Server is responsible for sending the information and graphics over the network to the client computer. NX Server supports most of the best-known Linux operating systems such as Red Hat, SUSE, Mandriva, Fedora Core, Debian GNU and Ubuntu. Solaris SPARC which is Unix operating system is also supported.

There are several server type possibilities for the NX Server depending on the means of usage. NoMachine offers many different versions of the server program to fill the various needs of the different sized companies. The versions that can be chosen are NX Free Edition, NX Enterprise Desktop Server, NX Small Business Server, NX Enterprise Server and NX Advanced Server. In all of the server choices, some functional characteristics are the same for all. These ones are listed below.[10]

- Deploys RDP and VNC desktops securely over the Internet
- Deploys X11 desktops securely over the internet
- Integrates single X11 applications with the native client desktop
- Lets user leave their applications running while disconnected and resume
- Lets applications access any file system on the client as if it were on the server
- Lets user cut and paste between the local and the remote applications
- Prints from the server to printers installed on the client
- Plays sound and the multimedia produced by the remote application
- Supports industry standard Secure Sockets Layer (SSL) encryption and Transport Layer Security (TLS)
- Integrates with SSH, avoiding the need for a further network access point
- Supports only PAM based authentication
- Connects to XDMCP servers, Windows Terminal Services and Citrix MetaFrame
- Provides integration with Web portals and corporate Intranets with NX Web companion and/or NX Builder
- Offers administration by the NX Server Manager Web interface

There are several more characteristics under development, such as support for the MacOS X and Internet Protocol version 6 (IPv6). These have been promised for the new version 3. There are also differences between the different server types, such as the number of connections allowed at the same time. These differences are shown in the *table 1* below.[10]

Table 1: Comparison of the NX Server products

	Nx Free Edition	NX Enterprise Desktop Server	NX Small Business Server	NX Enterprise Server	NX Advanced Server
Simultaneous Connections	2	2	10	unlimited	unlimited
Number of Users	2	unlimited	10	unlimited	unlimited
Integrates with LDAP Authentication Infrastructure and Windows Active Directory Server	-	-	-	+	+
Supports Multi-Node Capabilities Balancing the Load Among the Available NX Nodes	-	-	-	-	+
Supports Automatic Provision for the "Guest" User and Session Run in Kiosk Mode	-	-	-	+	+
Offers Easy Administration by the NX Server Manager Web Interface	-	-	-	+	+

As can be seen from the *table 1* above, there are a lot of benefits from buying a license for NX Enterprise Server or NX Advanced Server. However these benefits are such that are not possibly needed in normal home use. In a bigger corporation there may be even hundreds of users, Microsoft Active Directory Server, Unix server with LDAP authentication, and even need for a public terminal running kiosk mode of NX Remote Desktop.If that is the case, NX Free Edition, NX Enterprise Desktop Server and NX Small Business server may not provide enough compatibility and options for use.

There is also an open source version of NX server, FreeNX. It is maintained and developed by a former NoMachine worker, Fabian Franz. It is a good choice for home usage, but there are also some problems that may prevent using it in a company environment. These problems, such as lack of support

and incompatible client versions are discussed in section 4.3. Since the server needs also a node module installed, it is studied next.

3.4. NX Node

Nodes can distribute the applications running in a session between different application servers. NX Node can also be used for defining what libraries and NX software executables user may use. In a NX Advanced Server Nodes can be also used to load balancing to achieve a faster connection, since many different routes from client to server can be used. Nodes can be installed in e.g. existing servers, so there is no need to acquire new computers. The NX Remote Desktop system needs also a client to establish a session with the server. Therefore it is examined in next section.

3.5. NX Client

The client is a end-user program through which the user establishes a connection to the server. NoMachine offers NX Clients for Linux, Microsoft Windows, Solaris, MAC OS X and embedded systems. The client login window can be seen on *figure 5*.

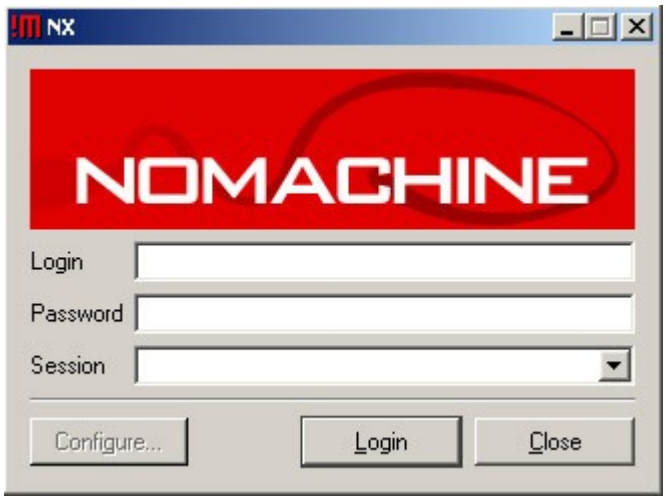

Figure 5: Client login window

The *figure 5* shows a client login window where login name, password and the session name created earlier is to be written. The client requests the server for username and password. Naming the session helps if there are lots of different connections to be made to different servers, or if one has to use different options such as display size or desktop manager.

3.6. Other NX Products

There are also some additional services that can be installed with NX Remote Desktop. NX Server Manager Beta is the upcoming product that can be used to tune the NX Server through a web interface. NX Web Companion Beta is a small Java applet and a plug-ins that allows one to run remote programs from corporate Intranet in a web browser. NX Builder Beta is a tool for defining and creating user sessions. This eliminates the need for a single user to configure the client, since all parameters are set by administrator. Since these services are still at the beta state, and they are not required in setting up the basic system, they are not tested or discussed further in this project.

4. NOMACHINE AND FREENX

In this section two possible manufactures of the NX Remote Desktop are studied. Overall information of both are included, and the comparison of the products are made in a mean of which is more suitable in this project. There are basically two authors for NX Remote Desktop technology, NoMachine and FreeNX. NoMachine is a company founded on 2003 and has been developing remote access and terminal services for Linux platform. The NX technology is General Public Licensed (GPL) which means that it is open source program. However the NoMachine products are commercial except for NX Free Edition and NX Clients. FreeNX provides a free server, but relies on the nodes and clients of NoMachine.

4.1. NoMachine

NoMachine is an Italian company founded on February 2003, and has become the most known NX technology vendor. The company provides a wide range of software filling the customers needs, from an individual users to the worlds biggest communication technology companies, such as IBM, Siemens and Nokia just to name a few. NoMachine offers a good customer support through their website even for a not-registered user. A registered user will be able to submit support enquiries to the NX support staff. In NoMachine.com web page there are documentation archives where one can find installation instructions Technical data, and frequently asked questions along with the web store can also be found.

NoMachine provides a wide range of NX Server products to fit different needs. There are versions for both, big corporations and home users. Depending on the server version even hundreds of users may use remote desktop simultaneously. The smallest version is called NX Server Free Edition which is free of charge and allows the maximum of two users connected to the server at the same time. Since there is also open source product available, it is described in the next section.

4.2. FreeNX

FreeNX is a project, which came up due to the free software nature of the NX technology. A former NoMachine worker Fabian Franz developes and maintains the program. As is with many of the open source projects where only a handful of people work with project in their spare time, updates are released rarely. FreeNX website offers almost no support, and all things related to NX technology are linked from the NoMachine website. The

FreeNX server is free, as the name says, but it needs NoMachines free client software, as well as nodes. The biggest problem between these two authors are that when NoMachine releases a new version of the client, it no longer works with the FreeNX server. This leads to a unwanted situation of using old and no more longer supported versions of the client program.

4.3. Comparison

This comparison is made by studying and testing more suitable version for Finnish Meteorological Institutes needs. Documentation, customer support and adaptability as well as security aspects are the most important questions. Since both of the products, FreeNX server and NoMachine NX server function in a similar way, the choice of which one to use depends greatly on the needs of the users and the environment where the server is to be installed.

If one supports Open Source products, and the security needs are not so strict, one may want to use FreeNX server. The drawback in this case is that NoMachine version of the node, and older version of client has to be used.. This is a possible security threat in a companies that try to avoid using old versions of programs, expecially when it comes to programs that operate through a network. A shortage of documentation and support need to be also considered in case something goes wrong.

Sharing files through Service Message Block (SMB) with computers running Windows and playing sounds from remote server are still under development in FreeNX. In the case of administrator usage, missing sounds are not concerned to be a problem. The missing support for Windows file sharing however is a problem since many companies including Finnish Meteorological Institute have a mixed environment of operating systems, and a lack of support for file sharing could prevent some work to be done.

If the security regulations are strict, as is in the case of Finnish Meteorological Institute, it would be wiser to use up-to-date clients using the products of NoMachine rather than FreeNX. Since the needs of companies

vary, NoMachine also provides a wide range of products from where users can find just the right product for their needs. In this study the chosen product is NoMachine NX Free Edition, which is free forever and supports two simultaneous users. This was chosen because the lack of support, shortage of documentation and incompatible client versions of FreeNX. The chosen product also serves the needs of this project because the main subject is to test how well this technology matches the LTSP administrators needs, not to supports tens or hundreds of users.

5. INSTALLING NX REMOTE DESKTOP

Installing the NX Remote Desktop to a server machine is rather easy, thanks to the good documentation of the NoMachine web site. The server computer can be an existing computer with Linux running on it, or it can be an individual computer with Linux acting only as NX Server. The task is quite straightforward, but a certain order in which the software components are installed have to be followed. The first one to be installed is the NX Client software, since the NX Server needs it. The second one is the NX Node, since the NX Server needs it also. The last one to be installed is the NX Server, and after that the basic connection could be set up. Since there are two versions of the server, NoMachine proprietary version and FreeNX free version, both installations are covered in this section.

5.1. Client Installation

The concerned version of NX client needs a package called **libstdc++2.10-glibc2.2_2.95.4-24_i386.deb**. This can be fetched via Ubuntu's official packet manager, apt-get. The second task is to download the client package from the NoMachine website. Depending on the platform, there are several

clients. Since the test system uses Ubuntu Linux, the packet to be fethched at the time of writing is **nxclient_2.1.0-9_i386.deb.** More up-to-date information about packages can be found on the web site http://packages.ubuntu.com

Installing the packages on the Linux system demands a superuser (root) privileges, though in Ubuntu this can be achieved by using the sudo. Sudo is a way to give a normal users some privileges that can be defined in advance.[11, p144]

There are also Windows client that can be used to set up a remote desktop connection to a Unix server. Windows client can be downloaded freely from the NoMachine's website, and the installation is very easy. After downloading the package, user has to just click the self extracting file to start installation. After selecting the destination where the client is to be installed, and choosing the start menu folder, and if the desktop icon is to be created, the client installs itself. After that the connection to the server is ready to be made.

5.2. Node Installation

After downloading the node package **dxnode_2.1.0-12_i386.deb** from NoMachine's website it just has to be installed with *dpkg* installation program. As the client, the node installation also needs superuser/root/sudo privileges. The client needs to be installed before node can be installed.[12]

5.3. Server Installation

After the client and node has been installed successfully, the server can also be installed. There are two options, either the NoMachine's version of the server, or the FreeNX server. If the NoMachine server is to be installed, the package **nxserver_2.1.0-13_i386.deb** can be downloaded from the vendors website.[13] After installation the system should be up and running.

If FreeNX version of the NX server is chosen, there are couple of things to do. At first the FreeNX repositories are added to the file /etc/apt/sources.list. The addresses to be added are

deb http://free.linux.hp.com/~brett/seveas/freenx dapper-seveas freenx
deb-src http://free.linux.hp.com/~brett/seveas/freenx dapper-seveas freenx

Since these packages can be GPG authenticated, the following command should be typed.

wget http://free.linux.hp.com/~brett/seveas/freenx/1135D466.gpg -O- | sudo apt-key add
After this the FreeNX can be installed with apt-get freenx command.

6. CLIENT SETUP

NX Client has a lots of options that can be set up. After some basic settings there are also advanced settings pages. This section will go through the various ways to set up a NX Remote Desktop connection, starting from the basic settings and after that explaining the advanced options that can be set up.

6.1. Basic Settings

After starting the client, the first page that appears holds the basic settings. First thing to do is to give the session a name. If one has several different client setups to different servers, it is recommended to give the session a describing name. By following the NX Connection Wizard all essential settings will be gone through. The first page of settings is shown on *figure 6.*

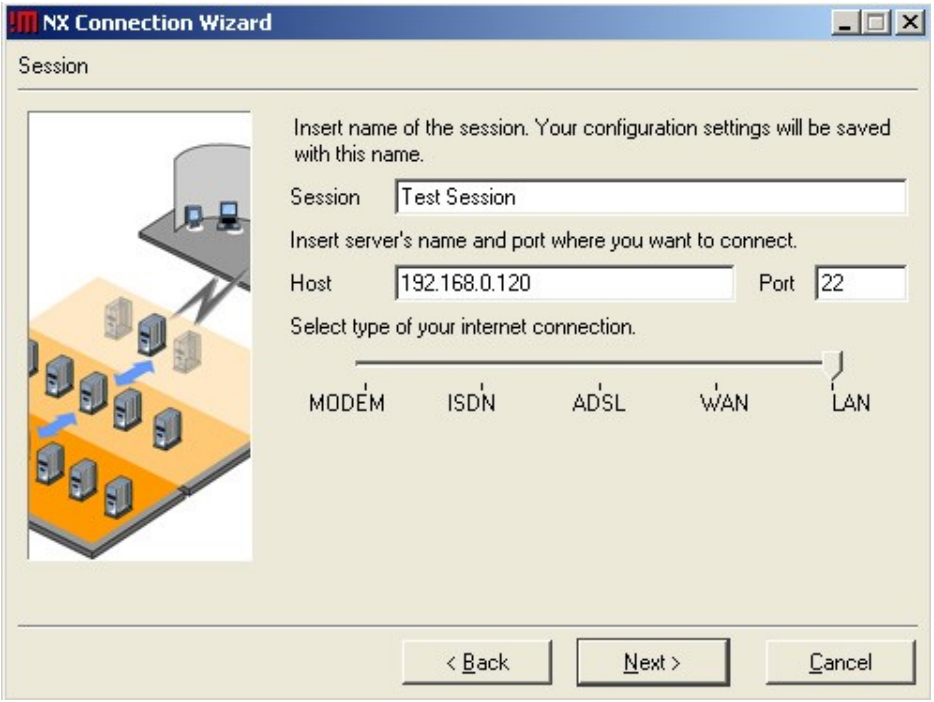

Figure 6: Settings page

Figure 6 shows that address of the host is typed in the *host* field along with the port that is listening the connections. The default port for the SSH is 22. The last thing to do is to define roughly the speed of the connection. In this case the speed is chosen to be LAN, because the tests are done in a local area network.

The next page is for defining which kind of connection is to be made. There are options to log on to Windows Terminal Server, or to use VNC remote

desktop. In this case the choice is to use Unix server with KDE desktop. Other desktops such as GNOME, CDE, XDM or even custom setting desktop can also be used, if they are supported by the operating system that runs NX Server. Finnish Meteorological Institutes LTSP system has both, KDE and GNOME installed, so the choice of KDE as a desktop is only a matter of opinion. *Figure 7* below shows the desktop settings page.

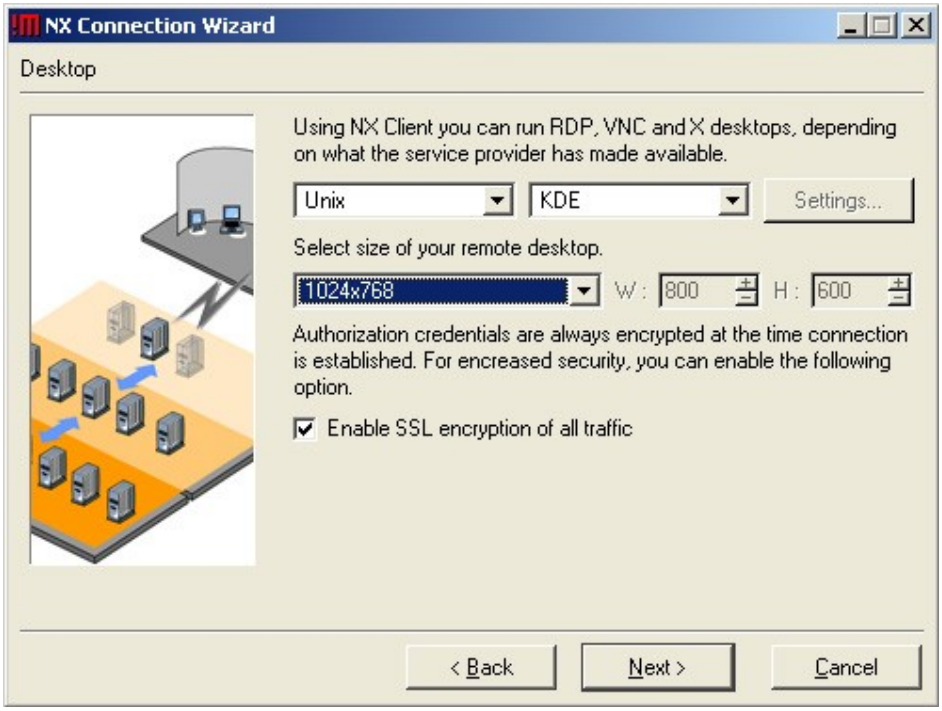

Figure 7: Desktop settings

As can be seen on *figure* 7 the second page of connection wizard manages the desktop settings of remote desktop connection. Screen size can be defined freely, starting from small custom sizes to a fullscreen mode. One can also enable SSL encryption for all traffic, which is strongly recommended because otherwise there will be data sent unencrypted. Without enabling SSL encryption NX Remote Desktop also uses random TCP ports that can cause problems with firewalls.

6.2. Advanced Options

After the basic settings are done, there is an option to go to change some advanced settings. The first interleaf that opens is general settings page. There it is possible to change the basic settings described in the previous section.

The next interleaf manages advanced options. There are options to disable no-delay on TCP connection. This causes the packets to be flushed on to the network more frequently. The second option is to disable ZLIB stream compression which is a lossless data compressing library that takes care of data compression. After that there is option to enable SSL encryption of all traffic, which was also on the basic settings. As informed before, it is wise to check this box. If the connection is done via HTTP proxy, one should type the address of the host and port on the next point of the settings page that can be seen on *figure 8* on the next page.

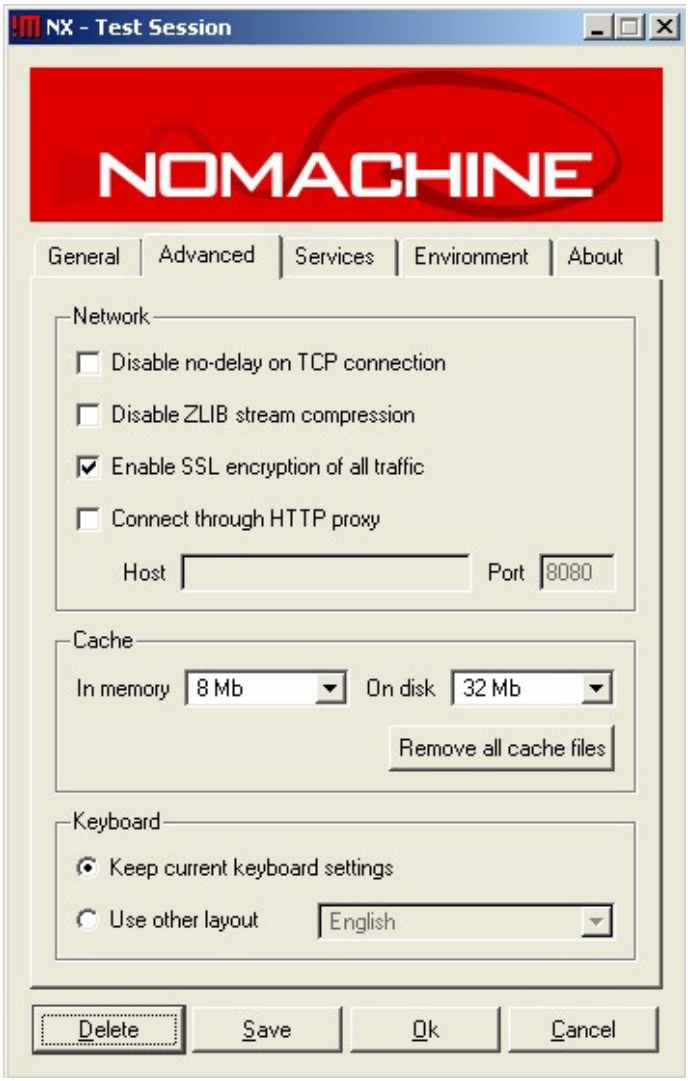

Figure 8: Advanced settings

As the *figure 8* shows, NoMachine client has an option for using cache. This makes it faster for example to re-open programs that were used before. Basic settings for cache are 8MB allocation in memory, and 32MB on the disk. One can alter these settings to fit one's needs depending of the computer in use. The last thing to do in this page is to define keyboard settings. If there are special needs to use other keyboard layout than the one in use, there are options to choose between many languages.

Figure 9 shows the *services* page of the client options. The options are explained below.

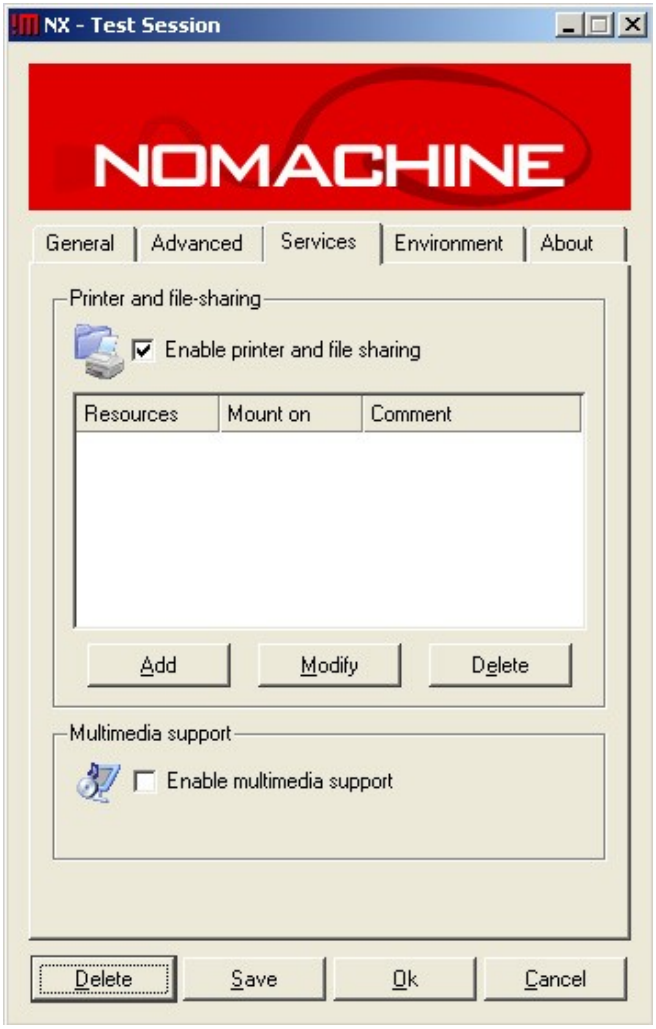

Figure 9: Services interleaf

On the services interleaf that can be seen in *figure* 9 it is possible to enable printer and file sharing. By pressing the add button the program opens a pop-up page, where shared resources are defined. Windows looks automatically for shared resources, folders and printers. In a Linux-version one has to enable SMB printing and file sharing, if such exist. There is also a possibility to enable CUPS printer sharing in Linux environment. On the bottom of the screen is an option to enable multimedia support on the services page, which will enable playing of sounds from the remote server.

The last essential page of options is called environment. It can be seen in *figure 10* below..

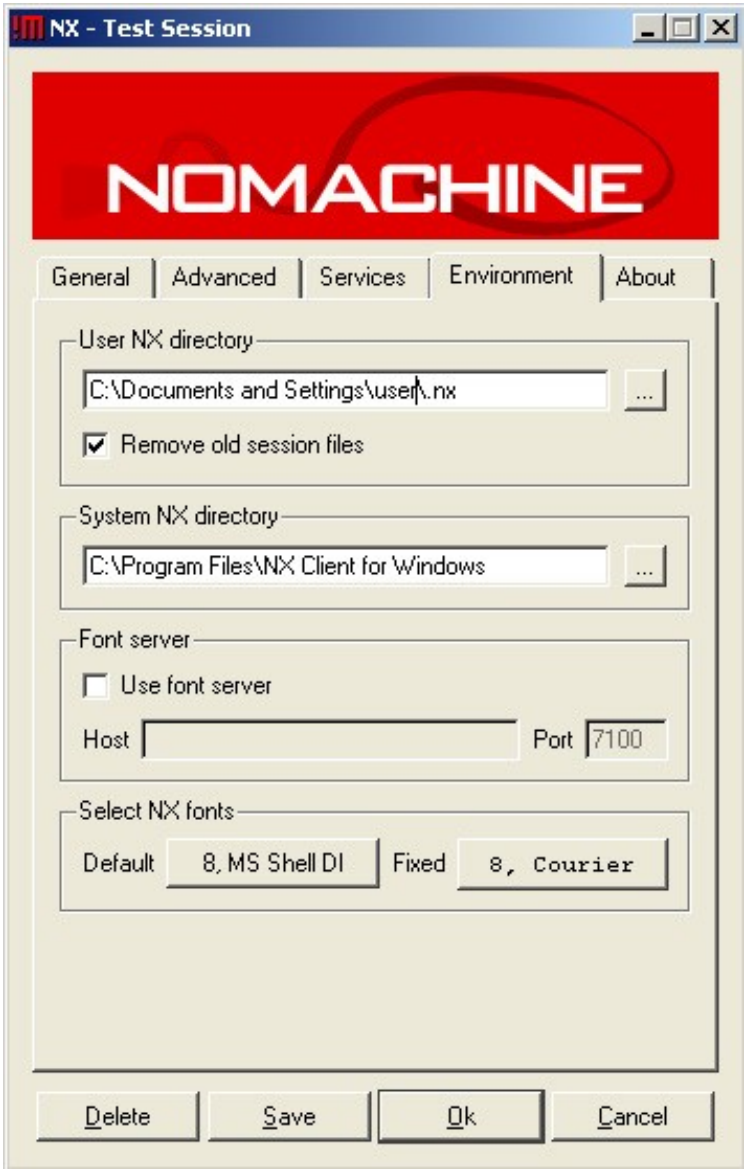

Figure 10: Environment interleaf

As can be seen on *figure 10* there is a possibility to change user's NX directory where all the user data of the session is stored. System NX directory can also be changed. The ones seen on the picture are Windows defaults. If there is a dedicated font server, one can check the box next to the text ”Use font server”. If checked, a hostname and a port have to be defined also.

When using Linux computer to create connection, defaults for the User NX directory and the System NX directory can be seen in a *figure 11* below. If using CUPS printer sharing, it is also necessary to define where the CUPS daemon is to be found, if not in default directory.

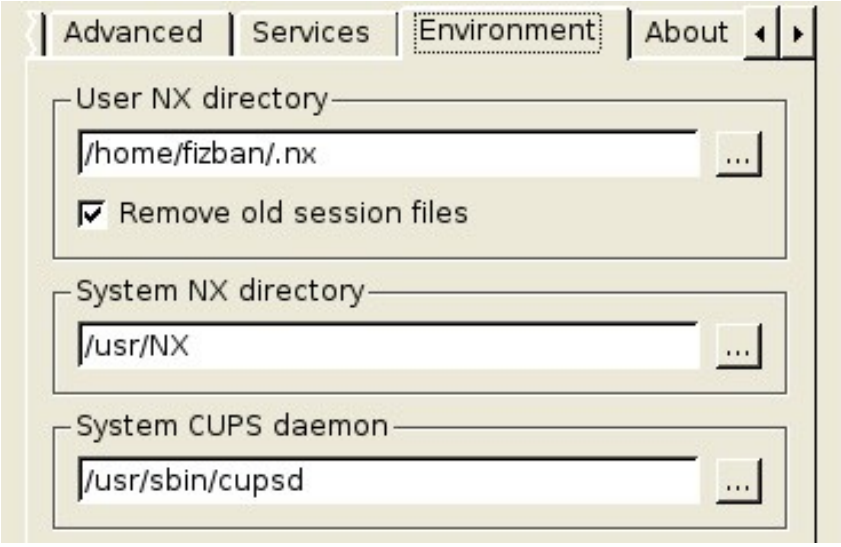

Figure 11: Default directory structure

As can be seen on *figure 11* the user directories in Linux environment differ slightly. The default for the user data directory is typical along with the Linux directory structure, but as can be seen, the NX Remote Desktop is installed default in the */usr/NX* directory instead of standard directory structure.

6.3. Recommended Options

There are some options that are recommended. Enabling SSL encryption for all traffic ensures that all phases of connection forming are encrypted. If the client party resides behind a HTTP proxy, it may be necessary to define a valid host for proxy, along with valid port number. One may also want to change the cache memory allocations to better suit for one's computer.

If network printers exists, they should be added to the services page in order for them to work. Sounds can also help, so enabling multimedia support on the same page is recommended.

7. TESTING

At first NX Remote Desktop was tested in a closed environment. Practically this meant testing the program at home. The network consisted of 3 computers, wired together through a simple 4-port hub. This of course created some collisions, but throughput was quite good. There were no single bottlenecks, since all of the network cards and hub were 100Mbit/s. Operating system in both servers was Ubuntu Linux 6.06, and the client computer was equipped with both, Ubuntu Linux 6.06 and Windows XP.

The network topology of testing environment can be seen on the *figure 12*.

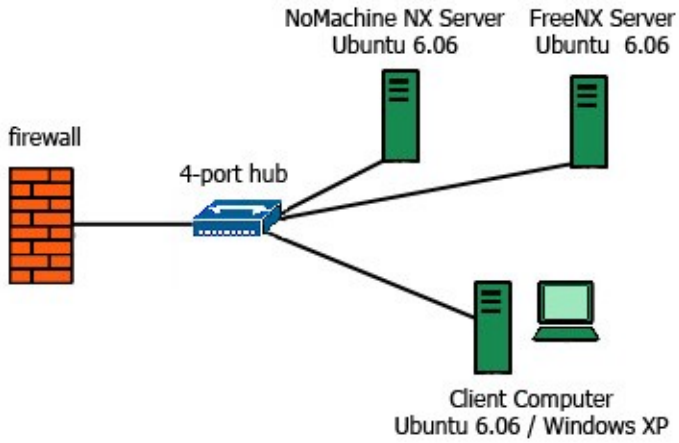

Figure 12: Test network

The test network seen on *figure 12* consists of three computers connected through a 4-port hub, and a firewall. Since the network is closed by firewall, there is no need for strict anti-virus policies which could lead to unwanted

blockings.

After the first installation when both server versions, NoMachine server and FreeNX server were put on the same computer, problems occurred. Both of the servers used the same directory structure, and the one installed later overwrited the files in the installation directory. After that FreeNX server was installed to a laptop computer, and NoMachine NX Remote Desktop server was installed to a PC workstation The overall setup was two separate servers and the testing workstation. OpenSSH packet was also installed on both of the computers, since NX connection relies on SSH technology. Packet updates were also done to server computers running Ubuntu, just to make sure everything is up-to-date.

7.1. Testing in a Closed Environment

After some complications described above, it was time to set up the remote desktop connection from the computer using Windows XP as the operating system. The client that NoMachine provides is very simple and straightforward to use, and using existing username and password, remote desktop connection was established with a default properties.

Since at least the NoMachine website promises that NX Remote Desktop will run over the average of 20Kbit/s connection, this was put in to test.[14] A freeware program called NetLimiter was installed on a Windows computer, and NX Client was allocated for 20Kbit/s bandwidth. There had also been some writings about Flash movies not working well. This was tested by watching YouTube videos remotely. Usability of office program, OpenOffice was also tested along with some basic games coming with Ubuntu. Shared printer feature was not tested.

7.2. Results

Connection to the NX Server was first tested with 20Kbit/s bandwidth allocation. This was to be sufficient enough by NoMachine web article. This took nearly twelve minutes until the remote desktop was usable. This is by no means a reasonable time for any kind of remote desktop connection to set up. The second thing was to double the bandwidth. When allocated 40Kbit/s, setting up lasted over five minutes. Using a shareware program called NetLimiter it was easy to increase bandwidth little by little, and at the same time calculate how long it takes before NX Remote Desktop is in full use. The statistics about starting time in function of time is shown on a *figure 13* below.

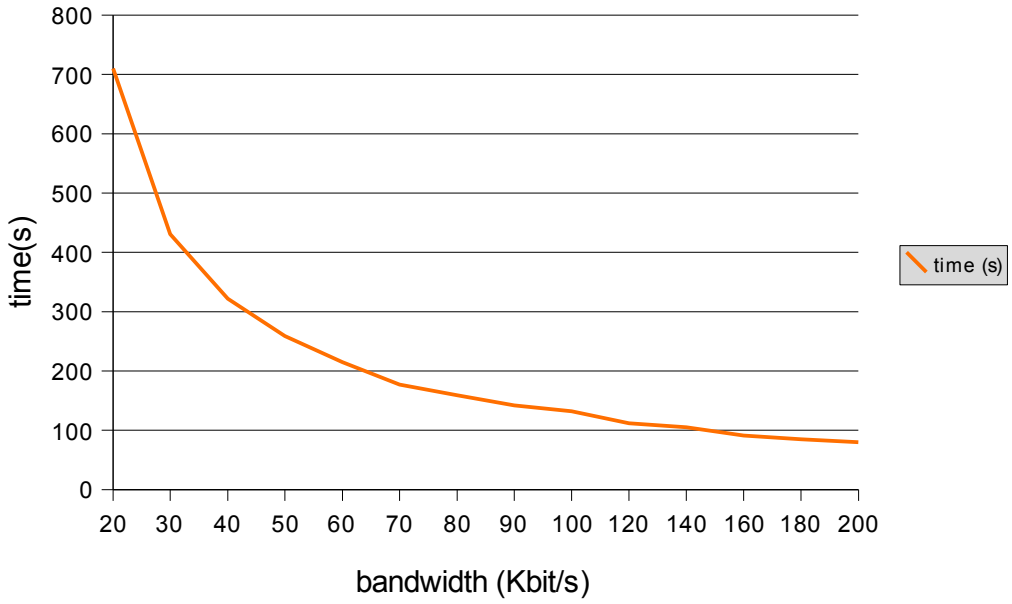

Figure 13: Starting time

As can be seen on a *figure 13* starting time with 20Kbit/s is nearly twelve minutes. By adjusting slowly the limit of the bandwidth, starting times began to come more realistic. The time was measured by digital clock, starting at

the time connection was started and ending when K-menu of the KDE desktop could be opened.

By studying statistics, it seems that in the connection phase, NX Remote Desktop uses from 300Kbit/s up to 900Kbit/s bandwidth. Under these circumstances it takes only under a minute to set up the connection. The *figure 14* on the next page shows how much NX Remote Desktop's SSH agent uses bandwidth when using several programs.

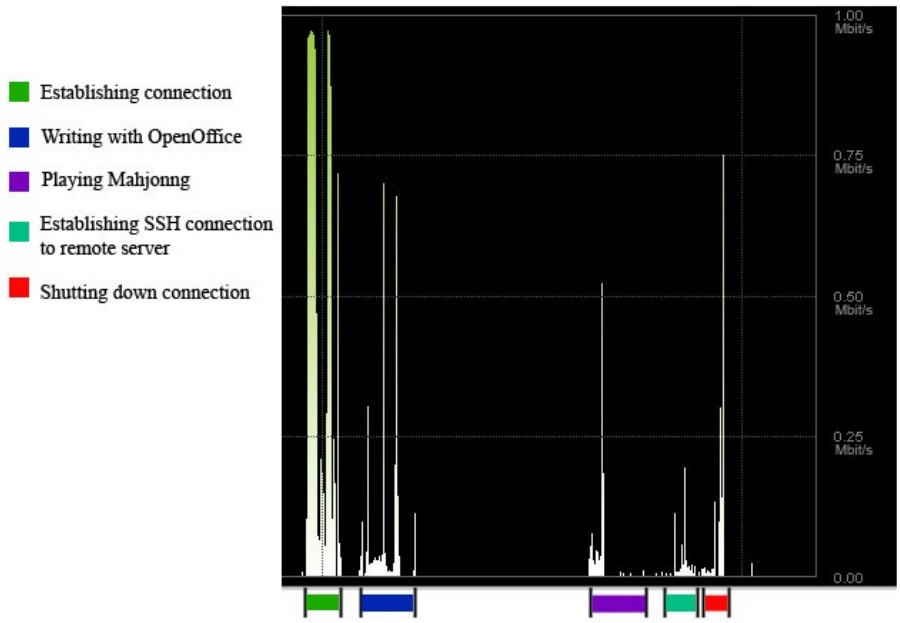

Figure 14: Bandwidth usage

As the *figure 14* shows, after the desktop is at use, need for the bandwidth greatly decreases as there is no need to move graphical data. Using OpenOffice shows a few peaks in the graphics, but these are mostly caused by creating new document. In a unlimited connection there is also a peak when playing Mahjonng and setting up a SSH connection to a remote server. Shutting the connection also adds a 750Kbit/s peak at most.

Flash movies started to load, but only scattered and distorted frames were shown. OpenOffice programs took a little time to start, but after that they run surprisingly well. Basic Internet browsing was also done, along with using basic programs and games that come with Ubuntu. There was a small lag when starting various programs in those cases when the bandwidth was limited under 100Kbit/s, but these lags disappeared after the first start of the concerned program.

7.3. Conclusions

It is remarkable how much the starting times differ from the ones promised on the NoMachine website. Even if there is a promise that NX Remote Desktop will work on an average of 40Kbit/s line, there are no notations about the fact that in that case it takes over five minutes to start. Even after that the usability is questionable. Although that the 40Kbit/s line is not the same as average of 40Kbit/s line, it is misleading. When using over 100Kbit/s line, the operability of the program increases. Launching an OpenOffice session took under 30 seconds, and there were substantial decreasing of lag. When the limits were taken off, NX Remote Desktop caused almost 1Mbit/s traffic, and this is by no means a light use of network bandwidth.

As predicated, Flash movies did not work well, there were only scattered frames and lots of delay. Sounds did not work either, even though the multimedia support was turned on. According to Ubuntu forums it was more of a Ubuntu's problem rather than NX's. OpenOffice was operating well, as well as other basic programs that were tested. Resuming the previous session was also working well, so there is no fear of losing one's jobs if network problems appear.

Overall the NX Remote Desktop is easy to install, and seems very usable at least with fast connections. On the other hand it was quite illogical that the

server needed client and node packages first in order to start install. In an closed environments such as local area networks and home networks NX Remote Desktop may well prove itself a useful tool for a remote desktop use.

8. FINAL SETUP AT FINNISH METEOROLOGICAL INSTITUTE

After the tests that were done in a home network, the final installation and testing will be done at Finnish Meteorological Institute. Because of different setting, including several firewalls and subnetworks with different policies the first thing to do is install the NX Remote Desktop server, client and node in a virtual server. This is done because the risk that something goes wrong in the installation or testing is far too great to be taken in the production environment. The virtual server is identical to actual LTSP servers, and resides in the same place as a network topology point of view, so if everything works as predicted, NX Server has just to be installed on the actual LTSP server. Testing is first done in an intranet, and after that by tunneling SSH connection through firewall, simulating situation where administrator is establishing connection from home.

8.1. Setting Up the System

At first a virtual server was put up, and Ubuntu 6.06 was installed on it. This was just a precaution that protected the production environment. The detailed installation instructions for server, node and client were covered in a section 5, and they will not be discussed here. All of the modules installed without problems, even though the CUPS printer share made error logs about not finding the server.

8.2. Testing

Testing was done with laptop computer using both, Ubuntu 6.06 and Windows XP as a operating system. At first connection was tested with Windows XP. Since there exist a couple of different firewalls in Finnish Meteorological Institutes network structure, problems were likely to appear. After couple of testings with different error messages from client it was time to check the firewalls. There were no indications in the firewall logs about that firewalls were blocking the connection. The next thing to do was to shut down anti-virus client After that connection was made successfully.

The second thing was to start testing SSH tunneling. Since NX Remote Desktop client has no built-in tunneling mechanism, tunnel had to be made by hand. Windows XP relies on third party software in putting up a SSH tunnel, so testing was made with a laptop running Ubuntu 6.06. Since the case was to test could the connection be established from anywhere, an ADSL line was used as a Internet connection and to make sure that firewalls treated the traffic properly. The SSH tunnel was made with command *SSH -l userid -L 7676:testserver.fmi.fi:22 firewall.fmi.fi*

Where *testserver* was the simulated LTSP server, and *firewall* was border firewall. After the tunnel has been set up, the NX client was told to establish connection to host *localhost* by using port *7676*. The tunnel worked fine, and remote desktop connection was made with no problems. The logs of the firewall also showed that only one SSH connection were running to the server as predicted.

9. RESULTS AND ANALYSIS

Results about the whole project is discussed here. Pros and cons of the NX Remote Desktop are studied, and the project itself is also analysed about how well it reached the goals that were set beforehand. The facts about how well does the NX Remote Desktop system perform in LTSP system administration is also discussed.

9.1. Overall Results

As there are several different remote desktop technologies existing at the same time, the results of this project were not predicted to be something shocking or revolving. The NX Remote Desktop is however very quick to set up, and easy to use which makes it notable candidate when choosing a remote desktop program. There were two different server types that were tested, FreeNX server and NoMachine NX server. Choosing the NoMachine's version was quite easy, since FreeNX is updated rarely, and it relies on NoMachine client. NoMachine also provides customers with customer support and offers also a free of charge version for limited use. There are lots of new promised features coming from NoMachine, but at the time they are still under development.

Since this project was about how well does it work for administrative use, *NX free forever* version was chosen, which makes it possible for two users to use remote desktop connection at the same time. The installation procedure itself was very quick. After founding out that anti-virus software needs tuning if NX Remote Desktop is taken in to use, everything worked well.

9.2. Conclusions

Experiences from working with NX Remote Desktop are promising. The program itself is very easy to install, despite the awkward procedure itself. Usually server programs are self-standing, but in the case of NX the server also needs node and client programs to be installed. In small home or company environments where there usually are not many firewalls or strict anti-virus programs NX Remote Desktop is very quick and easy to install and a good choise for accessing other computers. In situations, like Finnish Meteorological Institute, where there are couple of different firewalls managing both Internet and LAN traffic, some tuning need to be made with anti-virus software when using Windows XP. Connections made from Ubuntu laptop worked fine all the time. Since the NX Remote Desktop connection can be tunneled through a SSH tunnel, firewalls caused no problems. One has just to ensure that the user has rights to access the firewall through which the connection has to be established, and that the firewall knows about the server where NX Remote Desktop Server is installed.

The bandwidth demands were quite suprising compared to announcements in NoMachine website. Working with modern desktop environments demand more bandwidth than predicted. It is possible, but not recommended to establish a connection if the line is less than 80Kbit/s. In a LAN setting this is usually not the problem, but with genuine use of remote desktop this has to be noted. Modern cellular phones with GPRS or 3G transmission techniques can easily be used for setting up the connection, but working through outdated modem lines can be nerve wrecking.

NoMachine's version of NX is easy to use, and author provides detailed documentation in NoMachine website. New features are under develompent at the time of writing, and it is possible that NX Remote Desktop will gain more attention in the future.

REFERENCES

[1] Ilmatieteen laitos - Organisaatio - Historia, *Ilmatieteen laitoksen historiaa,* (Dec 21) [WWW-document] http://www.Finnish Meteorological Institute.fi/organisaatio/historia.html (Accessed Jan 7 2007)

[2] Aaltonen, Kimmo, Finnish Meteorological Institute, *Re:IL:n IT-infrasta* [e-mail message] receiver Poskiparta, Simo, (Sent Jan 5, 2007)

[3] Lintu, Veli-Matti, Opinsys: *Järjestelmän kuvaus* (2006) [Opinsys Intranet-document] (Accessed Feb 22, 2007)

[4] Figure 1, *The LTSP network* (2007) [According to Opinsys Intranet document] (Accessed Feb 22, 2007)

[5] Network Working Group, *The Secure Shell (SSH) Protocol Architecture Phaseout*, (Jan 2006) [WWW-document] http://tools.ietf.org/html/rfc4251 (Accessed Jan 11, 2007)

[6] Citrix Systems, *Understanding Citrix ICA*, (2007) [WWW-document] http://www.citrix.com/English/ps2/products/qa.asp?contentID=186&faqID=5638&title=Understanding+Citrix+ICA (Accessed Feb 13)

[7] Nemeth, Evi, Snyder Garth and Hein, Trent R, (2006) *Linux Administration Handbook*.2nd edition. Prentice Hall

[8] The Official Red Hat Linux Reference Guide, *section 10. Layers of SSH Security,* (2001) [WWW-document] http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/ref-guide/ch-ssh.html (Accessed Jan 11, 2007)

[9] Figure 4, *The SSH tunnel* (2007) [According to Nemeth, Evi, Snyder Garth and Hein, Trent R, (2006) *Linux Administration Handbook*.2nd edition. Prentice Hall]

[10] NoMachine NX - Products Page, *Features and Requirements*, (2007) [WWW-document] http://www.nomachine.com/features.php (Accessed Mar 3, 2007)

[11] Hill, M.B, Bacon, Jono, Burger, Corey, Jesse, Jonathan, Krstić, Ivan, (2007) *The Official Ubuntu Book, section 4: Advanced Usage and Maintainint Ubuntu.* Prentice Hall

[12] Regis, Silvia, NoMachine: *NX Node Installation Instructions*, (Apr 24, 2006) [WWW-document] http://www.nomachine.com/documentation/node/install.pdf (Accessed Jan 13, 2007)

[13] Regis, Silvia, NoMachine: *NX Server Installation Instructions*, (Apr 24, 2006) [WWW-document] http://www.nomachine.com/documents/server/install.pdf (Accessed Jan 13, 2007)

[14] NoMachine NX – Support: Article, *#AR12B00119*, (2007) [WWW-document] http://www.nomachine.com/ar/view.php?ar_id=AR12B00119 (Accessed Mar 6, 2007)

```
# Set config file format version.

#
CONFIG_FILE_VERSION = "1.0"

#
# Set the log level of NX Server. NX server logs to the syslog all
# the events that are <= to the level specified below, according to
# the following convention:
#
# KERN_ERR         3: Error condition.
# KERN_INFO        6: Informational.
# KERN_DEBUG       7: Debug-level messages.
#
# Note, anyway, that NX server uses level 6 in the syslog to log the
# event. This is intended to override any setting on the local syslog
# configuration that would prevent the event from being actually logged.
#
# The suggested values are:
#
# 6: This is the default value. Only the important events
#    are logged.
#
# 7: Sets logs to level debug.
#
#SESSION_LOG_LEVEL = "6"

#
# Specify hostname for the NX server.
#
#SERVER_NAME = "localhost.localdomain"

#
# Specify the TCP port where the NX server SSHD daemon is running.
#
#SSHD_PORT = "22"

#
# Set the base display number for NX sessions.
#
#DISPLAY_BASE = "1000"

#
# Set the maximum number of displays reserved for NX sessions.
#
#DISPLAY_LIMIT = "200"

#
# Set the maximum number of concurrent NX sessions.
#
#SESSION_LIMIT = "20"

#
# Set the maximum number of concurrent NX sessions that can be run by
# a single user. By default a user can run as many sessions as they
# are allowed on the server. By setting this value to 1, the system
```

```
# administrator can force the user to terminate any suspended session
# before starting a new one.
#
#SESSION_USER_LIMIT = "20"

#
# Set for how long NX server will retain data related to terminated
# sessions in its session history.
#
# <0: Never delete data from NX session history.
#
#  0: Disable NX sessions history.
#
# >0: Keep data in session history for this amount
#     of seconds.
#
# The default value, 2592000, lets NX server keep session data
# for 30 days.
#
#SESSION_HISTORY = "2592000"

#
# Allow the server to terminate oldest suspended sessions:
#
# 1: Enabled. Enable the automatic kill of the suspended
#    sessions.
#
# 0: Disabled. Suspended sessions are never terminated.
#
# When this option is set, and the server capacity has been reached,
# i.e. the maximum number of concurrent sessions could be exceeded,
# the server will kill the oldest suspended sessions to make room for
# the new user.
#
#ENABLE_AUTOKILL_SESSIONS = "0"

#
# Enable persistent sessions for users. If the option is followed by
# the keyword 'all', all users are allowed to run persistent sessions.
# Alternatively, it can be followed by a list of comma-separated user-
# names. The default value is 'all' which corresponds to enabling
# persistent sessions for all users. Values specified are overriden
# by the value set for the 'DISABLE_PERSISTENT_SESSION' key.
#
#ENABLE_PERSISTENT_SESSION = "all"

#
# Disable persistent sessions for users. If the option is followed by
# the keyword 'all', no user is allowed to run persistent sessions. Al-
# ternatively, the option can be followed by a list of comma-separated
# usernames. The default value is the empty string which corresponds
# to disabling persistent sessions for no user. The values specified
# override the values set for the 'ENABLE_PERSISTENT_SESSION' key.
#
#DISABLE_PERSISTENT_SESSION = ""

#
# Enable or disable SSL encryption of all traffic.
```

```
#
#
# 1: Enabled. Unencrypted connections between the proxies will
#    be allowed.
#
# 0: Disabled. Forbid the use of unencrypted connections. The
#    server will force the client to tunnel the proxy
#    connections over the encrypted channel.
#
# Session negotiation always happens across an encrypted channel.
# Normally the user can specify if subsequent communication must
# take place through a direct connection between the proxies or by
# tunneling it through SSH. You may uncomment the following line
# and set the value to 0 to increase the security of the host
# server or if your NX server is behind a firewall preventing
# the access to the set of ports used by the NX server.
#
# Unencrypted sessions require that the firewall lets the proxies
# communicate over the TCP ports ranging from:
#
# DISPLAY_BASE + 4000
#
# to:
#
# DISPLAY_BASE + 4000 + DISPLAY_LIMIT
#
# By default the user is allowed to specify if a session will run
# unencrypted, for example when running the session across the
# same LAN or when performance is an issue.
#
#ENABLE_UNENCRYPTED_SESSION = "0"

#
# Enable or disable clipboard:
#
# client: The content copied on the client can be pasted inside the
#         NX session.
#
# server: The content copied inside the NX session can be pasted
#         on the client.
#
# both:   The copy&paste operations are allowed between both the
#         client and the NX session and viceversa.
#
# none:   The copy&paste operations between the client and the NX
#         session are never allowed.
#
#ENABLE_CLIPBOARD = "both"

#
# Enable or disable NX users DB:
#
# 1: Enabled. Only users listed in NX users DB can login to the NX
#    server.
#
# 0: Disabled. All the authenticated users can login.
#
# If the NX user DB is disabled, any user providing a valid password
```

```
# from local DB or through SSHD authentication, can connect to the NX
# system. This is likely to be the default when SSHD authentication
# with PAM is enabled.
#
#ENABLE_USER_DB = "0"

#
# Enable or disable NX password DB:
#
# 1: Enabled. Use NX password DB to authenticate users.
#
# 0: Disabled. Use SSHD + PAM authentication.
#
# System administrators can enable a restricted set of users to con-
# nect to the NX server by setting ENABLE_USER_DB to 1 and adding
# those users to the DB. If user is enabled to connect, his/her pass-
# word will be verified against the current PAM settings by the SSHD
# daemon.
#
# If both 'ENABLE_USER_DB' and 'ENABLE_PASSWORD_DB' are set to 0, any
# user being authenticated by SSHD account will be enabled to connect
# to the system.
#
ENABLE_PASSWORD_DB = "0"

#
# Specify hostname of the server used for NX SSH authentication.
#
#SSHD_AUTH_SERVER = "127.0.0.1"

#
# Specify the TCP port where the SSHD daemon is running on the NX SSH
# authentication server.
#
#SSHD_AUTH_PORT = "22"

#
# Enable or disable statistics:
#
# 1: Enabled. Enable the production of NX statistics.
#
# 0: Disabled. Disable the production of NX statistics.
#
# Run the nxstat daemon in background. This daemon can be used to
# produce statistics about either the host machine, by elaborating
# data provided by the nxsendor daemon, or the NX services. The NX
# statistics can be queried and displayed by the NX Server Manager.
#
#ENABLE_STATISTICS_DB = "0"

#
# Specify the hostname or IP address where the nxsensor daemon will
# be assumed to run.
#
#SERVER_SENSOR_HOST = "127.0.0.1"

#
# Specify the port where the server will contact the nxsensor daemon
```

```
# to collect the statistics.
#
#SERVER_SENSOR_PORT = "19250"

#
# Enable or disable load balancing:
#
# 1: Enabled. Let  NX server decide the node host according to the
#    hosts available in the NX Node DB.
#
# 0: Disabled. Start NX session on localhost.
#
#ENABLE_LOAD_BALANCING = "0"

#
# Specify a list of comma-separated 'hostname:port' values for XDM
# server.
#
#ROUND_ROBIN_XDM_LIST = "localhost:177"

#
# Enable or disable the XDM round robin query:
#
# 1: Enabled. Let NX server decide XDM host according to hostnames
#    that are defined in the ROUND_ROBIN_XDM_LIST key.
#
# 0: Disabled.
#
#ENABLE_ROUND_ROBIN_XDM_QUERY = "1"

#
# Enable or disable the XDM indirect query:
#
# 1: Enabled. Let the user obtain a list of available XDM hosts.
#
# 0: Disabled.
#
#ENABLE_INDIRECT_XDM_QUERY = "0"

#
# Enable or disable the XDM direct query:
#
# 1: Enabled. Let client specify XDM host.
#
# 0: Disabled.
#
#ENABLE_DIRECT_XDM_QUERY = "0"

#
# Enable or disable the XDM broadcast query:
#
# 1: Enabled. Let client connect to the first responding XDM host.
#
# 0: Disabled.
#
#ENABLE_BROADCAST_XDM_QUERY = "0"

#
```

```
# Specify path and name of the command 'sessreg'.
#
#COMMAND_SESSREG = "/usr/X11R6/bin/sessreg"

#
# Specify the location and file name of the SSH authorized keys.
#
#SSH_AUTHORIZED_KEYS = "authorized_keys2"

#
# Accept or refuse the NX client connection if SSHD does not export
# the 'SSH_CONNECTION' and 'SSH_CLIENT' variables in the environment
# passed to the NX server.
#
# 1: Refuse. Check the remote IP and don't accept the connection
#    if it can't be determined.
#
# 0: Accept. Check the remote IP and accept the connection even if
#    the remote IP is not provided.
#
#SSHD_CHECK_IP = "0"

#
# Enable or disable support for automatic provision of NX guest users:
#
# 1: Enabled. The NX server creates system accounts for guest users.
#
# 0: Disabled.
#
#ENABLE_GUEST_USER = "0"

#
# Specify the base username to be used by NX server to create guest
# users accounts. The server will add a progressive number to the
# name specified by GUEST_NAME, according to the range of values set
# in the BASE_GUEST_USER_ID and GUEST_USER_ID_LIMIT keys.
#
#GUEST_NAME = "guest"

#
# Set the base User Identifier (UID) number for NX guest users.
#
#BASE_GUEST_USER_ID = "10"

#
# Set the maximum User Identifier (UID) number reserved for NX guest
# users.
#
#GUEST_USER_ID_LIMIT = "200"

#
# Set the Group Identifier (GID) for NX guest users. The specified
# GID must already exist on the system.
#
#GUEST_USER_GROUP = "guest"

#
# Set the maximum number of concurrent NX guest users.
```

```
#
#GUEST_USER_LIMIT = "10"

#
# Set the maximum number of NX sessions a NX guest user can run before
# his/her account is terminated.
#
#GUEST_USER_CONNECTION_LIMIT = "5"

#
# Set for how long the NX server has to retain NX guest users accounts.
#
#  0: NX guest users accounts are never removed.
#
# >0: Maintain NX guest users accounts for this amount
#     of seconds.
#
# The default value, 2592000, lets NX server keep guest users accounts
# for 30 days.
#
#GUEST_USER_ACCOUNT_EXPIRY = "2592000"

#
# Set for how long the NX server has to keep alive a NX guest user's
# session. When the time is expired, NX server will kill the session.
#
#  0: NX guest user's session is never terminated.
#
# >0: Keep alive NX guest user' session for this amount
#     of seconds.
#
#GUEST_CONNECTION_EXPIRY = "0"

#
# Enable or disable possibility for NX guest users to suspend sessions:
#
# 1: Enabled. The NX server lets NX guest users suspend sessions.
#
# 0: Disabled.
#
#GUEST_USER_ALLOW_SUSPEND = "1"

#
# Set the home directory for NX guest users. If an empty value is
# specified, the NX server will create the guest user's home in the
# default directory set on the system.
#
#GUEST_USER_HOME = "/home"

#
# Enable or disable removing the guest user from the system when the
# account is expired:
#
# 1: Enabled. When the guest account is expired, the NX server will
#    delete the account from both the system and the NX guests DB
#    and will remove the home directory.
#
# 0: Disabled. When the guest account is expired, the NX server will
```

```
#    keep the guest account on the system but will forbid this user
#    to start new sessions on the server.
#
#ENABLE_GUEST_WIPEOUT = "1"

#
# Allow the server to set disk quota for the guest accounts:
#
# 1: Enabled. When a new guest account is created on the system,
#    the server will set the disk quota for this user.
#
# 0: Disabled. No restrictions on the amount of disk space used
#    by each guest user are applied.
#
#ENABLE_GUEST_QUOTA = "0"

#
# Specify the username of the account to be used as a protoype for
# propagating its disk quota settings to all the new guest accounts.
# If the softlimit or the hardlimit on either the inode or the disk
# block are set, they will override the settings applied to the user
# prototype.
#
#GUEST_QUOTA_PROTONAME = "protoguest"

#
# Specify the maximum amount of disk space available for each of the
# guest users, checked as number of inodes. This limit can be exceeded
# for the grace period.
#
#GUEST_QUOTA_INODE_SOFTLIMIT = "0"

#
# Specify the absolute maximum amount of disk space available for
# each of the guest users, checked as number of inodes. Once this
# limit is reached, no further disk space can be used.
#
#GUEST_QUOTA_INODE_HARDLIMIT = "0"

#
# Specify the maximum amount of disk space available for each of the
# guest users, checked as number of disk blocks consumed. This limit
# can be exceeded for the grace period.
#
#GUEST_QUOTA_BLOCK_SOFTLIMIT = "0"

#
# Specify the absolute maximum amount of disk space available for each of the
# guest users, checked as number of disk blocks consumed. Once this
# limit is reached, no further disk space can be used.
#
#GUEST_QUOTA_BLOCK_HARDLIMIT = "0"

#
# Specify the grace period, expressed in seconds, during which the
# soft limit, set in the GUEST_QUOTA_INODE_SOFTLIMIT key may be
# exceeded.
#
```

```
#GUEST_QUOTA_INODE_GRACE_PERIOD = "0"

#
# Specify the grace period, expressed in seconds, during which the
# soft limit, set in the GUEST_QUOTA_BLOCK_SOFTLIMIT key may be
# exceeded.
#
#GUEST_QUOTA_BLOCK_GRACE_PERIOD = "0"

#
# Specify a list of comma-separated filesystem names or devices to
# which the disk quota restrictions will be applied. The default
# value is 'all' which corresponds to applying the disk quota limits
# to all the filesystems having disk quota enabled.
#
#GUEST_QUOTA_FILESYSTEMS = "all"

#
# Set the User Identifier (UID) number for NX users. If an empty value
# is specified, the NX server will create the account with the default
# value set on the system.
#
#USER_ID = "10"

#
# Set the Group Identifier (GID) for NX users. If an empty value is
# specified, the NX server will create the account with the default
# value set on the system.
#
#USER_GROUP = "users"

#
# Set the home directory for NX users. If an empty value is specified,
# the NX server will create the user's home in the default directory
# set on the system.
#
#USER_HOME = "/home"
```

```
#
# Set config file format version.
#
CONFIG_FILE_VERSION = "1.0"

#
# Set the log level of NX node. NX node logs to the syslog all the
# events that are <= to the level specified below, according to the
# following convention:
#
# KERN_ERR        3: Error condition.
# KERN_INFO       6: Informational.
# KERN_DEBUG      7: Debug messages.
#
# Note, anyway, that NX node uses level 6 in the syslog to log the
# event. This is intended to override any setting on the local sys-
# log configuration that would prevent the event from being actually
# logged.
#
# The suggested values are:
#
# 6: This is the default value. Only the important events
#    are logged.
#
# 7: Set the log level to debug.
#
#SESSION_LOG_LEVEL = "6"

#
# Enable or disable the automatic clean-up of NX session directories
# at the time sessions are terminated:
#
# 1: Enabled. This is the default value.
#
# 0: Disabled. Directories are prefixed by 'T-' and left
#    for further reference.
#
#SESSION_LOG_CLEAN = "1"

#
# Enable or disable NX node to log the X client stderr.
#
# 1: Enabled. The standard error of the X clients is redirected to
#    the 'clients' file in the session directory.
#
# 0: Disabled. The standard error of the X clients is redirected to
#    /dev/null.
#
#CLIENT_LOG = "1"

#
# Set the maximum size allowed for the log of the X clients. The node
# will terminate the session if this limit is exceeded. The default
# value is 4194304 bytes (4MB).
#
#CLIENT_LOG_LIMIT = "4194304"
```

```
#
# Set the maximum size allowed for the session log. The node will
# terminate the session if this limit is exceeded. The default value
# is 4194304 bytes (4MB).
#
#SESSION_LOG_LIMIT = "4194304"

#
# Enable or disable SSL encryption of all traffic between server and
# node.
#
# 1: Enabled. Unencrypted connections between the server and
#    the node will be allowed.
#
# 0: Disabled. Forbid the use of unencrypted connections. The
#    node will force the server to tunnel the proxy connections
#    over the encrypted channel.
#
# Session negotiation always happens across an encrypted channel.
# Normally the user can specify if the subsequent communication
# must take place through a direct connection between the proxies
# or by tunneling it through SSH. You may uncomment the following
# line and set the value to 0 to increase the security of the node
# host or if NX node is behind a firewall preventing the access to
# the set of ports used by the NX node.
#
# Unencrypted sessions require that the firewall lets the proxies
# communicate over the TCP ports ranging from:
#
# DISPLAY_BASE + 4000
#
# to:
#
# DISPLAY_BASE + 4000 + DISPLAY_LIMIT
#
# By default the user is allowed to specify if a session will run
# unencrypted, for example when running the session across the same
# LAN or when performance is an issue.
#
#ENABLE_UNENCRYPTED_SESSION = "1"

#
# Specify path and name of the command 'fuser' to identify processes
# using files or sockets.
#
COMMAND_FUSER = "/bin/fuser"

#
# Specify path and name of the command 'lsof' to list open files.
#
#COMMAND_LSOF = "/usr/sbin/lsof"

#
# Specify path and name of the command 'xauth' to edit and display
# the authorization information used when connecting to the X server.
#
#COMMAND_XAUTH = "xauth"
```

```
#
# Specify path and name of the command 'xmodmap' to edit and display
# the keyboard modifier map and keymap table.
#
#COMMAND_XMODMAP = "xmodmap"

#
# Specify path and name of the command starting 'CDE'.
#
#COMMAND_START_CDE = "cdwm"

#
# Enable or disable use of 'xkbcomp' command:
#
# 1: Enabled. Use 'xkbcomp' command.
#
# 0: Disabled.
#
#ENABLE_COMMAND_XKBCOMP = "1"

#
# Specify path and name of the command 'xkbcomp' to compile XKB key-
# board description.
#
#COMMAND_XKBCOMP = "xkbcomp"

#
# Specify location and file name of the keymap file used by 'xkbcomp'.
#
#XKBCOMP_KEYMAP_FILE = "/etc/X11/xkb/keymap/xfree86"

#
# Specify the location and file name of the SSH authorized keys.
#
#SSH_AUTHORIZED_KEYS = "authorized_keys2"

#
# Specify the font server to be used by NX agent. By default the NX
# agent only uses the X11 system fonts. Uncomment the following line
# to enable use of an X Font Server.
#
#AGENT_FONT_SERVER = "unix/:7100"

#
# Specify the path of default X window system startup script.
#
#DEFAULT_X_SESSION = "/etc/X11/xdm/Xsession"

#
# Set the default DPI of the X server to the specified value. This
# should normally not be required, but some desktop applications fail
# to set an appropriate value and fall back to 75 DPI, which is the
# value reported by default by the X server.
#
#DEFAULT_X_DPI = "96"

#
# Specify the path of libraries to be added to the agents environment.
```

```
# Be sure that NX libraries are listed first.
#
#AGENT_LIBRARY_PATH = "/usr/NX/lib"

#
# Specify the path of libraries to be added to NX proxy environment.
#
#PROXY_LIBRARY_PATH = "/usr/NX/lib"

#
# Specify a list of libraries to be preloaded in X applications. This key
# is only used when running sessions in single application mode without NX
# agent encoding. Starting from version 1.5.0, the node doesn't preload the
# NX X11 libraries.
#
# The default is an empty list.
#
# APPLICATION_LIBRARY_PRELOAD =
"/usr/NX/lib/libX11.so.6.2:/usr/NX/lib/libXext.so.6.4:/usr/NX/lib/libXcomp.so.1:/usr/NX/lib/libXcompext.so
.1:/usr/NX/lib/libXrender.so.1.2"
#
#APPLICATION_LIBRARY_PRELOAD = ""

#
# Specify a list of directories to be added to the library path of X
# clients when run inside a session in single application mode without
# NX agent encoding.
#
# The default is an empty list.
#
# APPLICATION_LIBRARY_PATH = "/usr/NX/lib"
#
#APPLICATION_LIBRARY_PATH = ""

#
# Enable or disable TCP_NODELAY setting in NX proxy.
#
# 1: Enabled. Let NX client choose whether to enable or not TCP_NODELAY
#    on proxy socket.
#
# 0: Disabled. Disable TCP_NODELAY.
#
# Due to a bug in old Linux kernels, enabling TCP_NODELAY when running
# sessions over PPP links can cause sessions to fail if a serious net-
# work congestion is encountered.
#
#PROXY_TCP_NODELAY = "0"

#
# Specify a list of comma-separated options to be added to the
# NX proxy transport. Look at the NX proxy documentation for more
# information about the available options. Options specified in
# 'PROXY_EXTRA_OPTIONS' override the parameters set by NX client.
#
#PROXY_EXTRA_OPTIONS = "limit=256k,link=modem"

#
# Append arguments to the command line used to run the NX agent for
```

```
# RFB (VNC sessions).
#
# Mutiple parameters can be specified by separating them with a blank
# character. Note that, for security reasons, no shell interpretation
# is made.
#
#AGENT_EXTRA_OPTIONS_RFB = "-viewonly"

#
# Append arguments to the command line used to run the NX agent for
# RDP (Windows sessions).
#
# Mutiple parameters can be specified by separating them with a blank
# character. Note that, for security reasons, no shell interpretation
# is made.
#
#AGENT_EXTRA_OPTIONS_RDP = "-m"

#
# Append arguments to the command line used to run the NX agent for
# X (Unix sessions).
#
# Mutiple parameters can be specified by separating them with a blank
# character. Note that, for security reasons, no shell interpretation
# is made.
#
#AGENT_EXTRA_OPTIONS_X = "-nocomposite -noshpix"

#
# Specify the default Window Manager to be used when running single
# applications in a virtual desktop.
#
#DEFAULT_X_WM = "twm"

#
# Specify the domain of the Windows Terminal Server.
#
#DEFAULT_RDP_DOMAIN = ""

#
# Specify the path of base directory where the NX node has to mount
# shares exported by the user. The default value is $(HOME)/MyShares.
#
#SHARE_BASE_PATH = "$(HOME)/MyShares"

#
# Allow the node to use privileged scripts to manage the shares:
#
# 1: Enabled. The node will use the suid scripts to mount and
#    unmount the client shares using the SMB or the CIFS file-
#    sharing protocol depending on which protocol is available
#    on both client and server side.
#
# 0: Disabled. The node will forbid any attempt to mount the
#    client shares.
#
#ENABLE_FILE_SHARING = "1"
```

```
#
# Specify the path of the directory holding CUPS binaries (f.e. the
# 'lpoptions' program).
#
CUPS_BIN_PATH = "/usr/bin"

#
# Specify the path of the directory holding CUPS programs and reserved
# for administrative purposes (f.e. 'cupsd' or 'lpadmin').
#
CUPS_SBIN_PATH = "/usr/sbin"

#
# Enable or disable CUPS support:
#
# 1: Enabled. Enable CUPS support.
#
# 0: Disabled. Disable CUPS support.
#
ENABLE_CUPS_SUPPORT = "1"

#
# Specify the file-sharing protocol to be used for attaching the
# filesystem to the target directory set by the SHARE_BASE_PATH
# key. The possible values are both,smbfs,cifs or none.
#
MOUNT_SHARE_PROTOCOL = "none"

#
# Specify the TCP port where the NX node SSHD daemon is running.
#
#SSHD_PORT = "22"

#
# Accept or refuse the NX client connection if SSHD does not export
# the 'SSH_CONNECTION' and 'SSH_CLIENT' variables in the environment
# passed to the NX node.
#
# 1: Refuse. Check the remote IP and don't accept the connection if it
#    can't be determined.
#
# 0: Accept. Check the remote IP and accept the connection even if the
#    remote IP is not provided.
#
#SSHD_CHECK_IP = "0"

#
# Enable or disable running nxsensor:
#
# 1: Enabled.
#
# 0: Disabled.
#
# Run the nxsensor daemon in the background. This daemon can be used
# to produce statistics about the node machine. Produced data is to
# be queried and elaborated by the nxstat daemon.
#
#ENABLE_SENSOR = "0"
```

```
#
# The hostname or IP address where the NX server will contact the
# nxsensor daemon to connect the statistics data. The key is also
# used by the nxsensor daemon to find out the network interface
# where it will listen for incoming connections.
#
#NODE_SENSOR_HOST = "127.0.0.1"

#
# The port where the NX server will contact the nxsensor daemon to
# collect the statistics data. The key is also used by nxsensor to
# find out the network interface where it will listen for incoming
# connections.
#
#NODE_SENSOR_PORT = "19250"

#
# If set, the specified message will be provided to the user if this
# is the first time the user starts a session on this node.
#
#NODE_FIRST_LOGIN_GREETING = "Welcome to your NX session"

#
# If set, the specified message will be provided to the user every
# time he/she has started a new session on this node.
#
#NODE_LOGIN_GREETING = "Welcome to your NX session"


#
# Specify path and name of the command to start the GNOME session
#
COMMAND_START_GNOME="/usr/bin/dbus-launch --exit-with-session gnome-session"

#
# Specify path and name of the command to start the KDE session
#
COMMAND_START_KDE="/usr/bin/dbus-launch --exit-with-session startkde"
```